УДК 681.03

ВИКОРИСТАННЯ ВІДЕОГРАФІЧНИХ ПРИСКОРЮВАЧІВ ДЛЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

А.Ю. Дорошенко, М.В. Котюк, С.С. Ніколаєв

НТТ України «КПІ» та Інститут програмних систем НАН України,
03187, Київ, проспект Академіка Глушкова, 40.
Тел.: 526 1538, dor@isofts.kiev.ua

Запропоновано нові методи та засоби високопродуктивних паралельних обчислень із використанням відеографічних прискорювачів компанії AMD/ATI та NVidia на основі платформи МИША та мов програмування МИША M10 і МАЯ.

The article describes new methods of parallel computing using AMD/ATI and NVidia graphics hardware and the MYSHA M10 science-oriented software development platform with MAYA E256 and MYSHA M10 programming languages.

Вступ

В останні роки спостерігається тенденція до значно швидшого зростання функціональності та продуктивності відеографічних прискорювачів (ВГП) персональних комп'ютерів у порівнянні з розвитком інших компонентів комп'ютерних систем. Спершу створені для візуалізації тривимірних зображень, відеографічні прискорювачі за останні роки у зв'язку зі збільшенням вимог до швидкості та якості візуалізації графічних зображень (в першу чергу внаслідок динамічного розвитку індустрії комп'ютерних ігор) отримали великий набір якісно нових функцій, що дає змогу використовувати їх як універсальні обчислювачі. Дослідницький напрямок використання відеографічних прискорювачів як універсальних обчислювачів (УО) отримала назву GPGPU (від англійського General-Purpose Computation Using Graphics Hardware).

Основними гравцями на арені GPGPU є найбільші виробники традиційних процесорів та ВГП – компанії Intel, AMD/ATI та Nvidia. Наукові дослідження в цій сфері проводять конструкторські відділи цих підприємств, багато компаній і університетів, що працюють і ведуть дослідження в галузі інформаційних технологій. В мережі Інтернет з 2004 року працює портал www.gpgpu.org, який є своєрідним «штабом» у цій галузі науки.

Першим загальнодоступним засобом GPGPU став ВГП GeForce 6 series від Nvidia з підтримкою технології Shader Model 3.0 (www.nvidia.com), для якого була створена високорівнева мова програмування шейдерів Cg (C for graphics) та інструментарій NVidia SDK для програмування ВГП і інтеграції з іншими технологіями програмування. Продовженням цієї інструментальної лінійки став комплекс CUDA (Common Unified Device Architecture) на базі мови Cg для ВГП нової архітектури GeForce 8 series та обчислювальних систем «Тесла» на основі таких ВГП. Компанія AMD пропонує спеціалізовані периферійні процесори FireStream, які за архітектурою нічим не відрізняються від масових «ігрових» ВГП серій Radeon X1000-X2000. Для програмування цих засобів також доступний відповідний інструментарій та розроблена в Стенфордському університеті бібліотека GROMACS (www.stanford.edu.net).

В цій роботі запропоновано нові методи та засоби високопродуктивних паралельних обчислень із використанням відеографічних прискорювачів компанії AMD/ATI та NVidia на основі платформи МИША та мов програмування МИША M10 і МАЯ [1–2].

Архітектура сучасних ВГП та програмна платформа МИША

Сучасні версії інтегрованих наборів системної логіки також оснащені потужними ВГП, потенціал яких може бути використаний для різноманітних обчислень. Відеографічний прискорювач – це спеціалізований периферійний процесор, призначений для візуалізації тривимірної графіки. Більшість ВГП реалізовані у вигляді плати, яка містить набір мікросхем пам'яті для збереження даних, які обробляються у ВГП (текстур, вершин тощо), з'єднаних з центральним процесором (ЦП) через периферійну шину та набір системної логіки. Проте існують такі ВГП, що використовують загальносистемну пам'ять, а також такі, що використовують і власну (на платі), і системну пам'ять (отримують доступ з допомогою драйвера).

Відеографічні процесори можна умовно розподілити на сім поколінь. До першого покоління можна віднести ВГП, що не передбачали жодних методів програмування, до другого – ВГП із можливостями конфігурування. Поява технології вершинних та піксельних шейдерів – мікропрограм, призначених для обробки елементів зображення та відповідних скалярних/векторних процесорів – у третьому поколінні ВГП поклала початок застосуванню ВГП для “неграфічних” обчислень. Водночас слід відзначити, що перша версія технології шейдерів Shader Model 1.0 (http://en.wikipedia.org/wiki/Shader) передбачала дуже обмежену функціональність, а тому про практичне використання ВГП третього покоління як багатоцільових обчислювачів говорити було зарано. Четверте покоління ВГП передбачало використання технології Shader Model 2.0, яка містить багато розширень, зокрема стали доступними умовні та безумовні переходи, а отже,

стало можливим керування ходом виконання програми. Саме ВГП четвертого покоління використовуються для обчислень в платформі МИША M10 [1]. ВГП п'ятого покоління передбачають підтримку технології Shader Model 3.0, яка забезпечує можливості створення програм-шейдерів із використанням широкого спектру операторів, зокрема, завантаження даних з відеопам'яті, динамічні (тобто, обчислені самим шейдером) переходи. Такі ВГП вже можуть використовуватися для розв'язування широкого кола задач. ВГП шостого покоління передбачають уніфікацію архітектури ядра графічного процесора, а також вдосконалення механізму виконання піксельних шейдерів. До цього покоління належать ВГП серії Radeon X1000 від AMD (www.amd.com). Проте за функціональністю шейдерної технології ВГП шостого покоління не відрізняються від ВГП п'ятого покоління. У ВГП четвертого-шостого покоління стали використовуватись масиви процесорів для виконання мікропрограм-шейдерів різних типів, перетворюючи ВГП таким чином у мультипроцесорні системи. Так, найпотужніший ВГП шостого покоління Radeon X1900 XTX містить 48 піксельних та 8 вершинних процесорів, що забезпечує теоретичну продуктивність 360 Гфлопс. І, нарешті, ВГП сьомого покоління можна вважати революційними: кожен такий ВГП є програмованою макроконвеєрно-потоковою машиною MIMD-архітектури.

Поява ВГП сьомого покоління пов'язують з випуском компанією Microsoft технології Direct 3D 10 (www.microsoft.com, www.msdn.ru). Застосована у ВГП сьомого покоління технологія Shader Model 4.0 передбачає скасування більшості обмежень, що існували в попередніх версіях. Сучасні ВГП сьомого покоління містять від 80 до 320 процесорів, що забезпечує гіпотетичну продуктивність одного ВГП до 520 Гфлопс, а сумарна гіпотетична продуктивність системи із центральним процесором Opteron QuadCore та 4 ВГП Radeon HD 2900 XTX від AMD становить 2,5 Тфлопс (www.amd.com). Таким чином системи відеографічних прискорювачів вже становлять реальну конкуренцію традиційним мультипроцесорним архітектурам за обчислювальною потужністю, а за співвідношенням ціна/продуктивність – значно їх переважають.

Специфіка «неграфічних» обчислень з використанням ВГП виключає можливість реалізації довільного алгоритму за допомогою звичних на «традиційних» обчислювальних засобів функцій. Проблема полягає у тому, що функції шейдера викликаються почергово для певного елемента у потоці даних для його видозміни. З іншого боку, програмна платформа МИША якраз зорієнтована на парадигму алгоритмічних обчислень із використанням потоково-функціонального програмування [1]. Зокрема, вона передбачає застосування розширеної технології обробки табличних і деревоподібних структур даних, і тому створення модулів виконання МИША-програм на ВГП виявляється нескладним завданням і дає змогу забезпечити ефективне використання ВГП як досить універсального обчислювача. МИША підтримує ефективний оптимізатор і технологію платформних побудов, що забезпечує поєднання крос-платформності із високою швидкістю виконання алгоритмів.

У контексті використання програмної платформи МИША для високопродуктивних паралельних обчислень з використанням відеографічних прискорювачів розглянемо архітектуру кластерних систем, методи інтеграції МИША з ВГП на різних рівнях та технології програмування алгоритмів з використанням мов МИША M10 та МАЯ E256 [2].

Інтелектуальний макроконвеєрний диспетчер

На підставі проведених досліджень було розроблено підхід «диференційованого кластера» для оптимізації алгоритмів для виконання на графічних прискорювачах та метод визначення ознак алгоритмів, які отримують максимальний виграш від виконання на відеографічних прискорювачах та різних видах процесорів. Головна перевага методу – можливість створення системи, яка дає сумарний ефект максимальної швидкості на усіх типах завдань за рахунок наявності у її складі процесорів, які оптимальні для певних видів задач. Це нагадує диференціацію клітин у вищих організмах, що в синергічному ефекті дає організму сукупну "функціональність" усіх диференційованих клітин, тоді як колоніальні організми, що складаються з "універсальних" клітин, перебувають на незрівнянно нижчому рівні розвитку та адаптації до навколишнього середовища. Слід вказати, що технологія диференційованого кластера кардинально відрізняється від ініціатив AMD Accelerated Computing та Intel Larrabee. Головна перевага технології диференційованого кластера полягає у її гнучкості та наявності універсальної перенацілюваної програмної платформи і орієнтації не на процесори, орієнтовані на розв'язання конкретних специфічних задач, а на адаптацію різних задач для процесорів різних архітектур, оптимальних для обчислення певної задачі.

Утворена в платформі МИША база евристик обсягом 65 МБ дозволяє обрати оптимальний обчислювач ще на стадії компіляції за рахунок розпізнавання подібних елементів алгоритмів. До того ж, до складу платформи МИША включено динамічний диспетчер завдань, який в ході виконання аналізує якість роботи алгоритмів і генерує відповідні інструкції вибору оптимального обчислювача. Диспетчер використовує декомпозиційно-орієнтовану архітектуру та принцип "найкоротшого" шляху доступу до даних. Після розподілу програми на окремі підалгоритми для кожного з них виконуються евристики-класифікатори, які дають змогу визначити можливість виконання даного алгоритму на різних видах процесорів. Евристики визначають гіпотетичні характеристики алгоритму, а потім для кожного алгоритму вираховуються індекси A, B і C. Індекс A – це показник залежності часу виконання певного алгоритму від вхідних даних; індекс B – це кількість алгоритмів, що містять звернення до даного алгоритму; індекс C – це індекс альтернативної вартості очікування виконання даного алгоритму відносно певного алгоритму X.

Розподіл алгоритмів виконується за пріоритетами відповідно до обраного оптимального обчислювача. Певна кількість обчислювачів системи (це змінюваний у залежності від експериментальних даних системи показник) прикріплюються до алгоритмів з найвищими пріоритетами, решта обчислювачів вважаються «плаваючими». Завдання, призначені для однорідних обчислювачів групуються за принципом найкоротшого шляху: тобто, завдання, що мають спільного «замовника», у найкращому випадку мають бути виконані на цьому самому вузлі. Евристичні алгоритми передбачають забезпечення «короткого шляху» для даних з врахуванням індексу С (відповідно, у випадку появи високопріоритетного завдання на певному вузлі обчислення не припиняються, якщо їх сукупний пріоритет за індексом В перевищує С-індекс «нового» завдання). Диспетчер за кілька експериментальних запусків визначає оптимальне розташування елементів системи, яке у випадку необхідності можна виправити вручну.

Динамічний диспетчер також використовує «плаваючу» систему розподілу даних на дискових накопичувачах окремих вузлів системи на базі «водопровідного» принципу Лєбєдєва. Однорідні завдання на основі евристичних алгоритмів диспетчер виконує на одному й тому самому обчислювальному вузлі. Це дає змогу не використовувати централізовані сховища даних, а зберігати необхідну інформацію на обчислювальних вузлах. Так, подібні та взаємопов'язані алгоритми часто обраховуються на окремих вузлах та відеоприскорювачах, що дозволяє мінімізувати час доступу до пам'яті за рахунок мінімізації звернень до зовнішніх інтерфейсів. Оскільки алгоритмами диспетчера є динамічні евристики, диспетчер може бути переналаштований на роботу і за іншими методами. Базовий системний алгоритм диспетчера орієнтований на експериментальне досягнення оптимальної продуктивності системи для кожного розраховуваного завдання. Розроблений диспетчер може використовуватись також для створення на базі МИША якісної платформ для віддалених розподілених обчислень (Grid).

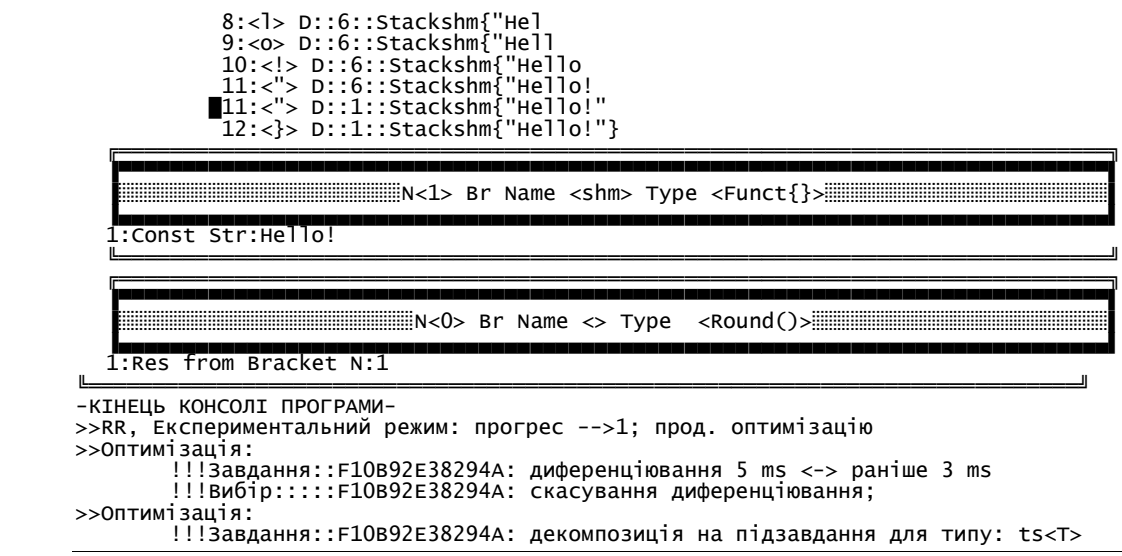
Особливості побудови ЕОМ на базі ВГП

Інтеграція відеографічних прискорювачів із платформою МИША можлива з використанням кількох методів на різних рівнях: інтеграцію на рівні базової апаратної платформи (при використанні ОС МИША для ВГП сьомого покоління), на рівні периферійного процесора (у ОС Windows) та на рівні графічних API Direct3D, OpenGL, CUDA. На різних рівнях використовуються інтеграційно-комунікаційні системи (ІКС), що забезпечують сумісність ВГП та можливість доступу до нього з боку платформи МИША.

Як вищезгадувалось, інтелектуальний диспетчер МИША використовує спеціалізовані алгоритми розрахунку оптимального розташування задач на обчислювачах. Дані алгоритми містять технологію передбачення руху даних обчислювальним конвеєром із використанням самонавчання. Диспетчер кластера для МИША був випробуваний на збудованій експериментальній системі ЦЕЗАР-1 (без використання ВГП) і показав середню ефективність використання ресурсів системи близько 70%. Цей показник був досягнутий завдяки якісним алгоритмам розпаралелення та декомпозиції процедур та технології швидкої маршрутизації даних у мережі. Проте у випадку використання ВГП необхідно виконувати додаткові розрахунки, адже потокова архітектура ВГП вимагає особливих принципів організації обчислень, малосумісних з макроконвеєрною архітектурою [3].

При розміщенні задач на ВГП МИША використовує два підходи – експериментальний та детермінований. Другий підхід дає змогу швидше отримати бінарний екземпляр програмної системи, здатний працювати на високій швидкості, проте може бути використаний лише на ВГП сьомого покоління. Експериментальний підхід універсальніший, проте вимагає кілька дослідних запусків для адаптації системи, при чому перші 2-3 запуски система працюватиме з дуже малою швидкістю. Проте даний підхід дозволяє максимально завантажити ВГП, а також керувати розміщенням обчислювальних задач на процесорах вручну в режимі реального часу. МИША інформує про свою роботу за допомогою інформаційних файлів та консолі, наприклад:

```
>>Пуск системи:::::::::ОЧІКУВАННЯ
Отримано завдання: F10B92E38296ED, 1398 команд
       !!!Готово – завдання ід-код F10B92E38294A, 6 наборів даних
       !!!Індекс A-10; BI-72;
>>Продовження::::::::::ОЧІКУВАННЯ
Отримано завдання: 5102D348224A45, 23 команди
       !!!Готово – TASK 5102348C294A, 2 наборів даних
       !!!Індекс A-10; BI-15;
       =->FLCA: DO NOW
>>Компіляція ОК::::::::::::Оптимізовано 4,1 %
>>Композиція ОК::::::::::::Пуск
>>Пристрій “Периферійний процесор на базі ВГП: PCIE-GF8800-MN8A8”
>>Інформує про готовність
>>Час розрахунку 5 мс
-КОНСОЛЬ ПРОГРАМИ-
╔═══════════════════════════════════════════════════════════════════════════════╗
Program Started :: 0s D::1::Stack
           0:<s> D::1::Stack
           1:<h> D::1::Stacks
           2:<m> D::1::Stacksh
           3:<{> D::1::Stackshm
           4:<"> D::1::Stackshm{
           4:<"> D::6::Stackshm{"
           5:<H> D::6::Stackshm{"
           6:<e> D::6::Stackshm{"H
           7:<l> D::6::Stackshm{"He
```




```
-КІНЕЦЬ КОНСОЛІ ПРОГРАМИ-
>>RR, Експериментальний режим: прогрес -->1; прод. оптимізацію
>>Оптимізація:
        !!!Завдання::F10B92E38294A: диференціювання 5 ms <-> раніше 3 ms
        !!!Вибір:::::F10B92E38294A: скасування диференціювання;
>>Оптимізація:
        !!!Завдання::F10B92E38294A: декомпозиція на підзавдання для типу: ts<T>
```

Важливим компонентом, що впливає на швидкість роботи систем із використанням ВГП, є шина зв'язку між ЦП та ВГП. У системі МИША важливість цієї шини зростає, адже між ВГП та ЦП безперервно передається величезний обсяг даних. Сучасні версії шин PCI Express (швидкість до 4 Гб/с у режимі використання 16 ліній) задовольняють вищенаведені вимоги, проте ВГП, що використовують менший обсяг каналів зв'язку (наприклад багато ВГП середнього цінового діапазону використовують 4 лінії), можуть працювати не на повну потужність внаслідок обмеженої пропускної здатності шини зв'язку. Водночас у робочих станціях із роз'ємами PCI Express Open X4 можна використовувати ВГП середнього цінового діапазону для повного використання потенціалу платформи. Очевидно, що архітектура ЦП та обчислювальної системи в цілому має відповідати принципу максимальної пропускної здатності, при якому у випадку архітектури AMD+DDR2 (Socket F):

– потоки “ВГП-ЦП” та “процесор-пам'ять” не перетинаються, що покращує можливості паралельної роботи компонентів системи;
– якісніше реалізована робота з оперативною пам'яттю (інтегрований контролер);
– ефективно розв'язана проблема комунікації між процесорами та ВГП та процесорів між собою з допомогою шини HyperTransport;
– легко реалізується масштабованість у рамках одного вузла, можлива архітектура із поєднанням окремих структурно незалежних підвузлів з допомогою шини HyperTransport.

Останній висновок був експериментально підтверджений у процесі досліджень платформи МИША в умовах практичного використання.

Важливим аспектом даної розробки є нова ідеологія націленого виконання обчислень на ВГП [4]. Враховуючи загальну орієнтацію платформи МИША на інтелектуальну автоматизовану конвертацію алгоритмів у процесі компіляції на цільовому обчислювачі, було розроблено багатоступеневу модель обчислень, яка дає змогу через кілька стадій поєднати макроконвеєрно-декомпозиційну модель, що використовується в оптимізаторі МИШІ, із сценарною векторно-піксельно-текстурною моделлю, що використовується у ВГП. Такий перехід з однієї системи в іншу дає змогу досягти оптимального поєднання ефективності та швидкості проектування алгоритму (алгоритм проектується з орієнтацію на абстрактний обчислювач гібридної рекурсивно-макроконвеєрної архітектури [5]) з високою продуктивністю роботи обчислювача.

Отже, набір завдань об'єднується в пакет, який формується за допомогою алгоритму “динамічного конвеєра із затримкою”. В пакет можуть входити як однотипні, так і різнотипні завдання, проте диспетчер намагається групувати однотипні завдання з урахуванням проблеми сумісності з ВГП четвертого, п'ятого та шостого покоління із обмеженнями на кількість команд у мікропрограмах-шейдерах. Подальша поведінка диспетчера залежить від типу ВГП та параметрів роботи. У випадку, якщо диспетчер містить достатню для завантаження ВГП кількість однотипних (тобто, на базі однієї функції) завдань у пакеті або наявна велика кількість завдань у черзі, пакет відправляється на кодогенерацію, а потім на виконання (у мультипроцесорних системах для кодогенерації виділяється окремий ЦП). Якщо ж однотипних завдань недостатньо, на ВГП шостого та сьомого покоління використовується тег-селективний алгоритм пакування різнотипних завдань. На ВГП попередніх поколінь завдання або очікують, або відправляються на кодогенерацію в неповному вигляді й залежності від пропозиції нових завдань, попиту на результати вже отриманих завдань і рівноважної ціни очікування однотипних завдань/виконання пакету в даний момент. При цьому у багатьох випадках використовується селективно-ковеєрний підхід на базі «водопровідного» принципу Лєбєдєва.

Як вищезгадувалося, кодогенерація для ВГП в платформі МИША динамічна, орієнтована на мультиядерні ЦП та багатопроцесорні обчислювачі. Виграш від ефективності обчислень та розширення функціональності і спектра можливих застосувань ВГП у прикладних задачах значно перевищує втрати від використання динамічної кодогенерації (а у багатопроцесорних системах ці втрати взагалі не помітні, адже для динамічної

кодогенерації використовується системно-орієнтоване ядро, на якому виконується диспетчер та операційної системи МИША 1.0).

Кожен окремий алгоритм, виділений протягом декомпозиції, на стадії компіляції виконуваного файла перекладається в проміжний асемблероподібний код скалярного процесора МП-7Ю, який за допомогою конвертора може бути швидко перетворений у виконуваний код для різних обчислювальних архітектур та різних наборів команд. При цьому застосовані алгоритми, що дозволяють перетворювати скалярні команди для виконання на векторних процесорах. Згенерований проміжний код зберігається у списку платформних побудов, а під час виконання завантажується залежно від профілю обладнання електронно-обчислювальної машини. Технологія профілів забезпечує зменшення витрат на оптимізацію в експериментальному режимі за рахунок пакетів евристик, апробованих на певних платформах. Слід також зазначити, що МИША дозволяє оптимізувати алгоритми для виконання на ВГП вручну.

Технологія ВГП-бібліотек стандартних алгоритмів

До складу платформи МИША входить багатоцільова алгоритмічна бібліотека із підтримкою використання ВГП як обчислювачів. Цей засіб дає змогу досягти дві мети – забезпечити високу швидкість роботи алгоритмів на ВГП та полегшити роботу програміста, використовуючи саме ті алгоритмічні засоби, які ефективно працюють із ВГП. Наприклад, для ЕОМ із використанням ВГП спроектовані окремі реалізації різних системних інфраструктур – технології еластичних масивів, деревоподібних структур, тунельно-індексного управління, фізичного моделювання, криптографії, архівації, динамічної кодогенерації, роботи з базами знань тощо. Ці технології мають окрему асемблерну реалізацію для ВГП різних виробників, що забезпечує і кросплатформність, і оптимальну швидкодію одночасно.

Ефективність роботи бібліотек МИША розглянемо на прикладі типових задач обробки баз знань (ці алгоритми мають бібліотечну реалізацію). Для дослідження використовувався фрагмент бази оптимізатора МИША, ЕОМ з ЦП AMD Athlon 64 X2 4800+, ВГП GeForce 8600 та Radeon X1600.

Платформа МИША при використанні бібліотеки демонструє таку продуктивність (рис. 1).

AMD Athlon 64 X2 4800+ ***Час роботи (секунд), БД обсягом 1024 К записів***

Задача з використанням БД	ЕОМ без ВГП, 2 ядра ЦП	-II- + 1 ВГП	-II- + 2 ВГП
Пошук	0,12	0,05	Нуль-час
Сортування	0,25	0,09	0,06
Множення	0,02	Нуль-час	Нуль-час
Нечіткий аналітичний розподіл	5,5	1,8	1
Виділення схожих об'єктів	12,3	2	1,5
Класифікація	1202	500	500
Індексація	8	2	0,87
Порівняння 2 БД	0,98	0,5	0,2

Рис. 1. Результати обробки бази знань за стандартними алгоритмами

При цьому аналогічні алгоритми, запрограмовані “вручну” на показники продуктивності, близькі до даних, виходять лише через 6-8 циклів оптимізації (рис. 2).

AMD Athlon 64 X2 4800+ ***Час роботи (секунд), БД обсягом 1024 К записів***

Задача з використанням БД	ЕОМ без ВГП, 2 ядра ЦП	-II- + 1 ВГП	-II- + 2 ВГП
Пошук	0,6	0,1	0,07
Сортування	0,5	0,22	0,1
Множення	0,02	Нуль-час	Нуль-час
Нечіткий аналітичний розподіл	10	3,3	2,4
Виділення схожих об'єктів	20,2	2	1,5
Класифікація	5125	602,8	594,5
Індексація	19,4	3,1	2,87
Порівняння 2 БД	5,6	2,1	1,3

Рис. 2. Результати обробки бази знань за складеними вручну алгоритмами мовою високого рівня

При цьому слід зазначити, що жоден інструмент програмування для ВГП не здатний виконувати всі необхідні операції над базами знань. На ряді задач роботи з БД знань МИШУ було порівняно з інструментарієм CUDA від NVIDIA (www.nvidia.com/cuda). Результати вказують значну перевагу динамічної технології перед статичною (рис. 3):

AMD Athlon 64 X2 4800+ ***Час роботи (секунд), БД обсягом 1024 К записів***

Задача з використанням БД	ЕОМ без ВГП, 2 ядра ЦП	CUDA	МИША
Пошук	0,6	Близько 0,9	Нуль-час
Сортування	0,5	Близько 0,3	0,06
Множення	0,02	Нуль-час	Нуль-час
Виділення схожих об'єктів	20,2	Близько 10	1,5
Порівняння 2 БД	5,6	Близько 5	0,2

Рис. 3. Порівняння продуктивності МИШІ (без ВГП) CUDA і МИШІ.(з ВГП).

Бібліотека МИША також передбачає набір функцій, які не впливають на швидкість роботи програмних засобів, проте вкладають думку розробника в таке річище, в якому він сам зможе обрати оптимальний для обробки оптимізатором МИША метод програмування універсальних алгоритмів. Оскільки використання цих засобів прискорює компіляцію та оптимізацію програмних засобів, спроектоване ПЗ отримує наступні дуже важливі властивості:

– відсутність потреби в адаптації на різні "стилі" роботи у випадку різкої зміни характеру оброблюваних даних (особливо це стосується систем, що обробляють динамічні деревоподібні структури даних);

– висока швидкість розгортання та відсутність проблем із активацією/розширенням гнучкої програмної системи внаслідок меншого попиту на послуги оптимізатора у алгоритмах, створених із використанням бібліотечних функцій.

Методи програмування ВГП засобами мови МИША М10

Мова програмування МИША передбачає кілька методів використання ВГП у різних видах алгоритмів. Більшість засобів базовані на стандартних засобах платформи, проте важливим елементом є структура мови та методи програмування, які вона заохочує.

Базовий метод передбачає використання ВГП в основних обчисленнях у алгоритмах обробки наборів даних (у контексті потоків ВГП). У мові програмування МИША акцент зроблено на обробку табличних та деревоподібних структур. Доступні в МИШІ засоби спонукають програміста використовувати цикли обробки пакетів даних сімейства **through,** наприклад, **through**, **through str**, **through ts**, **through ts str**. Як свідчить досвід, у програмах МИША ступінь використання цих циклів становить 55% від усіх алгоритмічних конструкцій (всього в МИШІ є більше 100 алгоритмічних конструкцій).

Даний метод програмування містить одну несподіванку: кількість циклів у програмі треба не мінімізувати, а максимізувати (звісно, в розумних межах). МИША містить інтегрований у систему розпаралелювання алгоритму модуль для визначення комплексу операцій над порцією даних і уніфікацією та перетворенням певного алгоритму, який проектувався без розрахунку на реалізацію мовою програмування МИША, на циклічно-потоковий. Зокрема, МИША передбачає можливість виконання різних частин різних функцій на різних видах обчислювачів – обирається найбільш ефективний варіант не лише для окремого алгоритму, і його частин, порівнюється виграш від виконання на оптимальному процесорі із втратами на пересилання даних. Наприклад, даний алгоритм ефективно виконуватиметься на ВГП за методом виділення частини підпрограми в окрему функцію:

```
through(c,ArTEC)
{
        if (ArTEC[c].TryP==CurentTECName)
        {
                VMAsmCnt=ArTEC[c].ExcP;
                err=0;
                VMDeep=1;
                TECerr=1;
                return;
        }
}
```

Аналізатор МИШІ виділяє та готує до виконання на ВГП цикли обробки пакетів даних сімейства **through** навіть у режимі роботи "без оптимізації", тобто цей процес виконується на рівні лексичного аналізатора структури алгоритму.

Мова МИША підтримує ряд додаткових властивостей, які не підтримуються платформою. Наприклад, в трансляторі передбачена технологія паралельної обробки в основній пам'яті системи та внутрішній пам'яті ВГП даних із використанням індивідуальних маркерів. Ця технологія дає змогу збільшити продуктивність та час обробки однієї порції даних у ВГП, мінімізувавши навантаження комунікаційні шини і основний процесор системи. Також є засіб для пакетної комунікації із ВГП (передбачений у ІКС для ВГП шостого та сьомого покоління.

Роботу оптимізатора МИША розглянемо на прикладі наступного алгоритму:

```
процедура Зшив(тс<тс<байт>> змінна і, ціл поч, ціл кін)
//> Зшив
{
     зростання(рах, поч, кін)
     {
        крізь(рах2, і[рах])
        {
      байт сб = і[рах][рах2];
      байт змінна сі = і[рах][рах2];
           сі<128?сі=0:сі=1;
           ціл пс = сі?255:0, пм = сб-пс;
           якщо(рах==і.ЛР()-1)
```

```
                {
                якщо(рах2!=i[рах].ЛР()-1) i[рах][рах2+1]+=пм*7/16;
                }
                     інакше
                {
                     i[рах+1][рах2]+=пм*5/16;
                     якщо(рах!=i[рах].ЛР()-1) i[рах+1][рах2+1]+=пм*1/16;
                     }
                  }
               }
            }
```

Вищенаведений алгоритм (в українському варіанті ключових слів МИША) реалізує апроксимацію 256-кольорового зображення у чорно-білий варіант за методом Error Diffusion [6].

Після розподілу алгоритму в трансляторі на окремі функції отримуємо наступний код мовою МИША (в трансляторі він представлений у вигляді деревоподібної об'єктної структури даних):

```
процедура $F10B92E38296ED(тс<тс<байт>>змінна $00,ціл $01,ціл $02,ціл $03,ціл $04)
{
                байт $L-00 = i[рах][рах2];
                байт змінна $L-01 = i[рах][рах2];
                     $L-01<128?$L-01=0:$L-01=1;
                     ціл $L-02=$L-01?255:0, $L-03 = $L-00-$L-02;
                     якщо($03==$00.ЛР()-1)
                {
                якщо($04!= $00 [$03].ЛР()-1) формула вгп мп $00[$03][$04+1]+=
                                                                    $L-03*7/16;
                }
                     інакше
                {
                     формула вгп мп  $00 [$03+1][$04]+= $L-03*5/16;
                     якщо($03!=$00 [$03].ЛР()-1) формула вгп мп $00[$03+1][$04+1]+=
                                                                    $L-03*1/16;
                     }
}

процедура $F10B92E38296ED(тс<тс<байт>> змінна $00, ціл $01, ціл $02, ціл $03)
{
          крізь($L-00, $00[$03])
          {
               вгп
               {
               $5102348C294A($00, $01, $02, $03, $L-00);
               }
          }
}

процедура Зшив(тс<тс<байт>> змінна і, ціл поч, ціл кін)
//> Зшив
{
       зростання(рах, поч, кін)
       {
          вгп
        {
        $F10B92E38296ED(і, поч, кін, рах);
        }
       }
 }
```

Після розподілу на підзадачі виконується генерування проміжного коду для алгоритмів та формул за двома профілями:

- для ВГП п'ятого та шостого поколінь використовується технологія «пакетних допоміжних обчислень»;
- для ВГП сьомого покоління підтримується селективно-алгоритмічна модель.

Розглянемо експериментально отримані результати про порівняльну швидкодію алгоритму Error Diffusion, реалізованого мовою МИША. При цьому для порівняння зазначимо, що на ЕОМ з одноядерним процесором без використання ВГП (Athlon 64 3200+) обробка зображень зайняла 4,5 сек. із мовою C++ (Intel C/C++ compiler) і 4,1 сек. (мова МИША M10) (рис. 4).

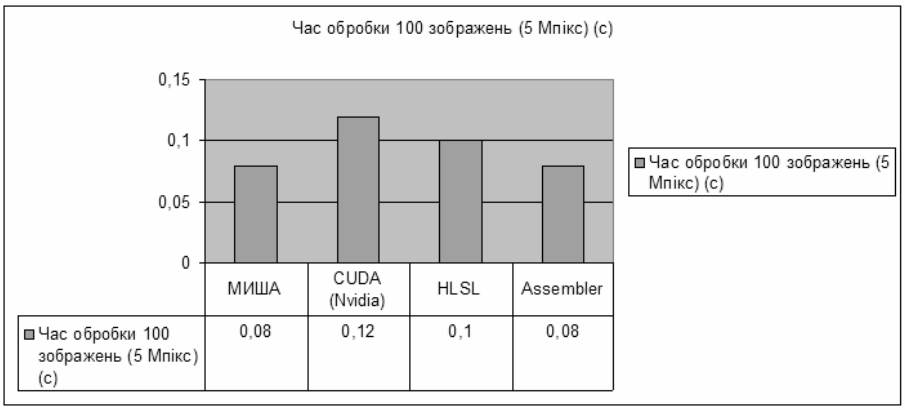


Рис. 4. Порівняння продуктивності технологій МИША, CUDA, HLSL, GPU Assembler

Мікропрограмно-компонентна модель мультиасемблерної мови програмування МАЯ

Якщо мова МИША M10 вимагає великої кількості аналізів та перетворень вихідного коду з метою забезпечення можливості виконання алгоритмів на ВГП, то призначена для розв'язання задач штучного інтелекту мультиасемблерна мова програмування МАЯ [2] на базі платформи МИША використовує мікропрограмно-евристичну модель, за деякими принципами дуже схожу на шейдерну технологію, використану у ВГП. Саме тому і виникла думка дослідити можливості використання мови програмування МАЯ при розв'язанні задач, зокрема і в сфері штучного інтелекту, із використанням ВГП.

Технологія МАЯ чітко структурована і має збалансовані характеристики. Мова МАЯ має декілька суттєвих переваг перед традиційними мовами програмування, закладених з самого початку розробки, а саме:

- МАЯ – це нова мова програмування, що не обтяжена проблемами сумісності із мовами-попередниками, а тому пропонує рафінований та однозначний і простий для розуміння як програмістом, так і інтелектуальними системами синтаксис, семантику і набір можливостей;
- єдність високорівневого та проміжного коду – мова просто не потребує транслятора з високорівневого у проміжний код;
- варіативна концепція (мова може бути як декларативною, так і імперативною у залежності від режиму використання);
- ефективна віртуальна машина на базі платформи МИША;
- можливість простої апаратної інтерпретації;
- компактність коду (алгоритми мовою МАЯ в середньому у вісім разів компактніші, ніж аналогічні алгоритми мовою С);
- евристична технологія програмування та запозичення нейронного і агентно-орієнтованого методів розробки;
- простота генерування коду та можливість самогенерації програм;
- відсутність потреби в об'єктній структурі алгоритму;
- динамічна технологія керування даними: прості та зрозумілі методи обробки графів, по суті, єдиного доступного типу даних;
- підтримка транснаціонального синтаксису на базі використання спеціальних символів замість варіативних наборів ключових слів (як це зроблено в мові МИША M10);
- можливість емуляції декларативних мов і перенесення коду такими мовами без змін.

Мова МАЯ використовує новітню мікропрограмно-евристичну технологію, яка пропонує повну абстракцію від архітектури конкретної ЕОМ і, в принципі, є оптимальним рішенням для створення перспективних програмних систем, передбачених В.М. Глушковим ще в 60-х роках [7]. Кожна програмна одиниця, створювана за допомогою МАЯ, є самостійним компонентом із власною інфраструктурою. У рамках технології використовується оновлена концепція об'єктно-орієнтованого програмування.

Рівень абстракції МАЯ передбачає використання віртуального простору, заповненого за деревоподібним принципом інтелектуальними компонентами-мікропрограмами, здатними до самонавчання. Ці компоненти мають власне сховище даних, сховище «попередніх реалізацій» (для відновлення у випадку дестабілізації) і службовий комплекс мікропрограм, які можна застосовувати для різних цілей – від керування автоматичною кодогенерацію до виконання допоміжних операцій (тобто, як підпрограми). Важливим нововведенням є дворівнева технологія самонавчання – на рівні алгоритму та на рівні ”констант” керування із використанням автоматичного збереження. Нова технологія дає змогу миттєво припиняти виконання МАЯ-програми і потім продовжити його, забезпечує дуже високу стійкість до пошкоджень таких програмних систем. У цьому плані, можна вважати, що МАЯ дозволяє моделювати мозкоподібні структури та нейронні мережі на якісно новому рівні, що пропонує широкі можливості для досліджень в галузі штучного інтелекту.

Проте МАЯ пропонує не тільки нову технологію розробки, а й високу швидкість виконання, зумовлену стрункістю методики розробки та широким використанням можливостей платформи МИША. Експлуатація інтегрованого середовища МИША-МАЯ показала велику кількість переваг комплексного застосування обох мов програмування в інтелектуальних системах.

Результати експериментальних досліджень

Проведені експериментальні дослідження складалися з трьох етапів – створення кластерної системи на базі локальної комп'ютерної мережі і дослідження розрахункових можливостей різних моделей ВГП, розробки низки науково-орієнтованих програмних систем та створення технологій пошуку, захисту та обробки інформації для масового користувача.

Кластерна система використовує диференційовану (гетерогенну) структуру. Вона складається зі з'єднаних у локальну мережу 100 Мбіт/сек. управляючого вузла та 11 обчислювальних вузлів із наступними характеристиками: ЦП Intel Pentium 4 3,0 GHz, 256 MB RAM, GPU Intel GMA 900 (DirectX9, Shader Model 2.0), НЖМД обсягом 80 Гб. Проведені вимірювання показали, що дана система має середню продуктивність на задачах, що не використовують ВГП, близько 50 млрд. операцій за секунду за умови використання технології МИША та максимальну досягнуту еквівалентну продуктивність на задачах із використанням ВГП близько 178,5 млрд. операцій за секунду. Отже, інтегровані ВГП п'ятого покоління, використані в системі, збільшують її продуктивність на певних задачах у 3 рази.

Висновки

Розглянуто методи використання відеографічних прискорювачів для науково-орієнтованих та інших обчислень, запропоновано нові підходи побудови кластерних систем із диференційованою структурою та організацією обчислень, описано методи використання платформи МИША для програмування обчислень на ВГП із використанням технології селективного, тунельно-індексного керування та методу допоміжних пакетних обчислень.

Описано нову нейронно-функціональну технологію програмування за допомогою мови програмування МАЯ та переваги даного підходу із використанням абстрактних обчислювальних модулів і середовища взаємодії при створенні програмних систем різного призначення.

Наведені результати експериментальних досліджень продуктивності платформи МИША свідчать про правильність обраного напряму створення автоматизованої платформи для паралельного програмування різних видів обчислювачів і вказують шляхи подальшого розвитку, а саме – розширення спектру драйверних інформаційно-комунікаційних систем для ВГП та вдосконалення евристичних алгоритмів формально-математичного (а не експериментального) визначення оптимального класу обчислювача для певних класів алгоритмів.

1. *Дорошенко А.Ю., Котюк М.В., Ніколаєв С.С.* Програмна платформа для наукових досліджень // Проблеми програмування. – 2007. – № 4. – С. 41– 52.
2. *Дорошенко А.Ю., Котюк М.В., Ніколаєв С.С.* Мультиасемблерна мова програмування для автоматизації наукових досліджень // Проблеми програмування. – 2008. – №1 (прийнято до друку).
3. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.*. Алгеброалгоритмические модели и методы параллельного программирования. – К.: Академпериодика, 2007. – 634 с.
4. *Дорошенко А.Ю., Куйвашев Д.В.* Інтелектуалізація перенацілюваної оптимізуючої компіляції для мікропроцесорів цифрової обробки сигналів // Проблемы программирования. – 2002. – № 1-2. – С.477– 488.
5. *Глушков В.М., Игнатьев М.Б.*, Мясников В.А., Торгашев В.А. Рекурсивные машины и вычислительня техника. – Киев, 1974. – 26 с. (Пр. пр. / АН УССР. Ин-т кибернетики; 74-57).
6. *Akhter S., Roberts J.* Multi-Core Programming. Increasing Performance through Software Multi-threading. – Intel Press, 2006. – 336 p.
7. *Глушков В. М., Ющенко Е.Л.* Теория распознавания образов и обучающихся систем // Техническая кибернетика. – 1963. – № 5. – С. 98– 102.